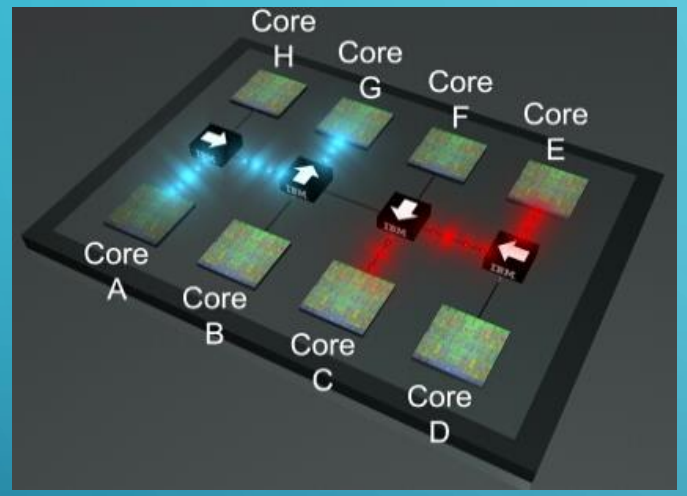
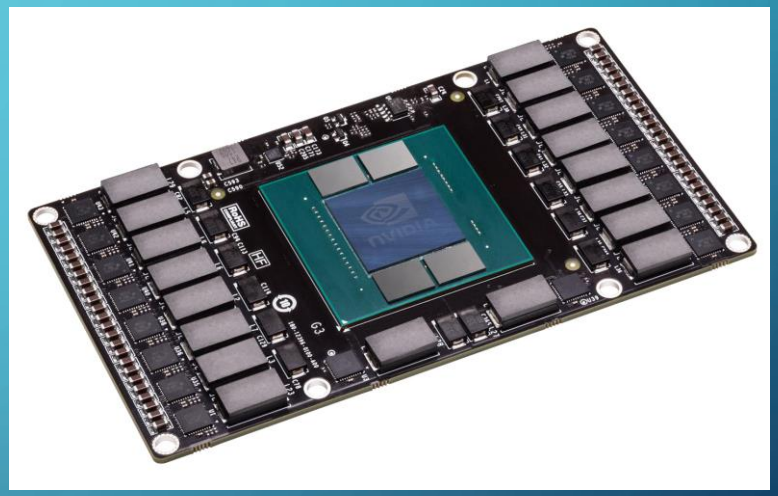
ПРЕОБРАЗОВАНИЕ ПРОГРАММ «РАСТЯГИВАНИЕ СКАЛЯРОВ»

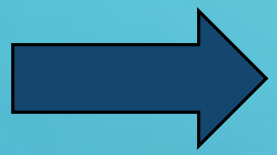
АВТОНОМОВ Д.А AVTONOMOVVV@GMAIL.COM

ШТЕЙНБЕРГ О.Б OLEGSTEINB@GMAIL.COM

ПАРАЛЛЕЛЬНЫЕ ВЫЧИСЛЕНИЯ

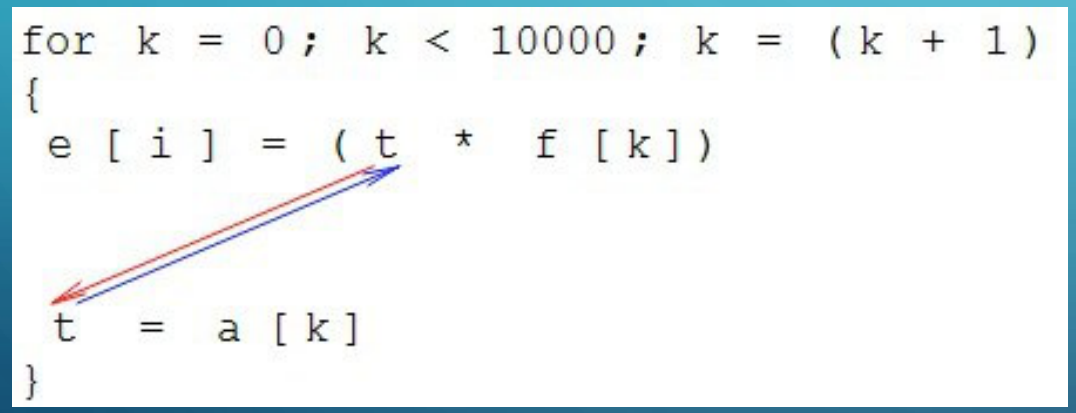




T  T_temp[i]

РАСТЯГИВАНИЕ СКАЛЯРОВ

- Растягивание скаляров - это преобразование цикла, способствующее распараллеливанию и векторизации, а так же разбиению цикла.



РАСТЯГИВАНИЕ СКАЛЯРОВ

- Растягивание скаляров - это преобразование цикла, способствующее распараллеливанию и векторизации, а так же разбиению цикла.

```
for k = 0; k < 10000; k = (k + 1)
{
 e[i] = (t * f[k])
 t = a[k]
}
```

```
int __uni1[10001];
__uni1[0] = t;
for (k = 0; k < 10000; k = k + 1)
{
  e[i] = __uni1[k] * f[k];
  __uni1[(k + 1)] = a[k];
}
t = __uni1[10000];
```

РАСТЯГИВАНИЕ СКАЛЯРОВ В ЛИТЕРАТУРЕ

The variable X in the following loop is an example.

```
      DO 3 I = 1, 10
      X = SQRT(A(I))
      B(I) = X
    3 C(I) = EXP(X)
```

In this loop, each occurrence of X can be replaced by $XX(I)$, where XX is a new variable.

Leslie Lamport «The Parallel Execution of DO Loops»

vector code. For instance, the loop

```
      do I=1,N
S1:     X = A(I) + B(I)
S2:     C(I) = X ** 2
      end do
```

can be vectorized by first expanding X into a temporary array, XTEMP

```
      allocate (XTEMP(1:N))
      do I=1,N
S1:     XTEMP(I) = A(I) + B(I)
S2:     C(I) = XTEMP(I) ** 2
      end do
      X = XTEMP(N)
      free (XTEMP)
```

and then generating vector code:

```
      allocate (XTEMP (1:N))
S1: XTEMP (1:N) = A(1:N) + B(1:N)
S2: C(1:N) = XTEMP(1:N) ** 2
      X = XTEMP(N)
      free (XTEMP)
```

Wolfe M. «ADVANCED LOOP OPTIMIZATIONS FOR PARALLEL COMPUTERS»

РАСТЯГИВАНИЕ СКАЛЯРОВ В ЛИТЕРАТУРЕ

- Что необходимо делать в случае присутствия рекуррентных операторов или условных операторов, не поясняется.

```
.....
for (int i = 0; i < 10000; i++)
{
  A[i] = T * C[i];
  T = T + Y[i];
}
.....
```

```
....
for (int i = 0; i < 100000; i++)
{
    If ( i % 100 == 0)
      {
        T = A;
      }
    C[i] = B[i] - T;
}
....
```

ALLEN, KENNEDY «OPTIMIZING COMPILERS FOR MODERN ARCHITECTURES»

- В центре их алгоритма стоит совокупность покрывающих генераторов, находимых с помощью SSA графа и графа потока управления

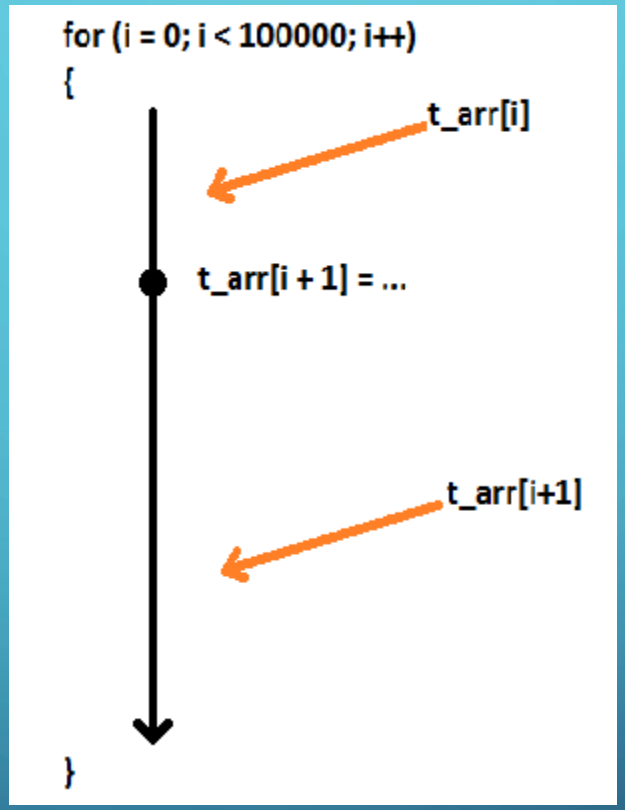
1. Create an array `T$` of appropriate length
2. For each S in the covering definition collection C, replace the `T` on the left-hand side by `T$(I)`.
3. For every other definition of `T` and every use of T in the loop body reachable by SSA edges that do not pass through S_0, the ϕ-function at the beginning of the loop, replace `T` by `T$(I)`.
4. For every use prior to a covering definition (direct successors of S_0 in the SSA graph), replace `T` by `T$(I-1)`.
5. If S_0 is not null, then insert `T$(0) = T` before the loop.
6. If there is an SSA edge from any definition in the loop to a use outside the loop, insert `T = T$(U)` after the loop, were `U` is the loop upper bound.

ОПЕРАТОР МНОЖЕСТВЕННОГО ВЫБОРА - SWITCH

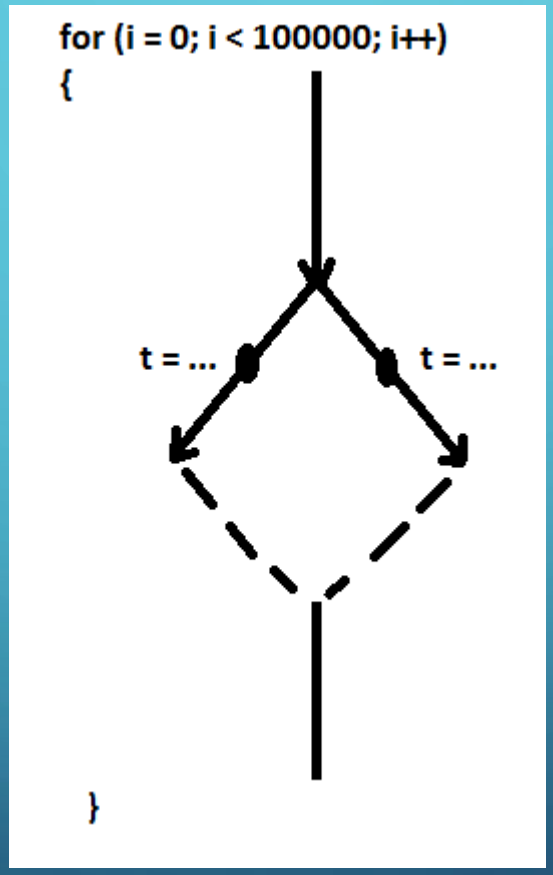
```
....
switch ( c )
    {
      case '1':
        T = ...;
      case '2':
        A[i] = C;
        break;
    }
....
```

```
for (i = 0; i < 100000; i++)
{
```

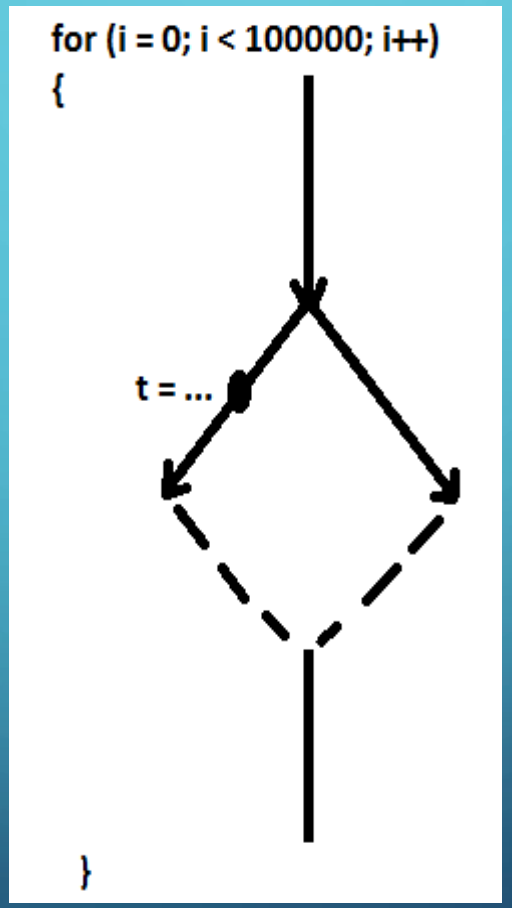

```
}
```

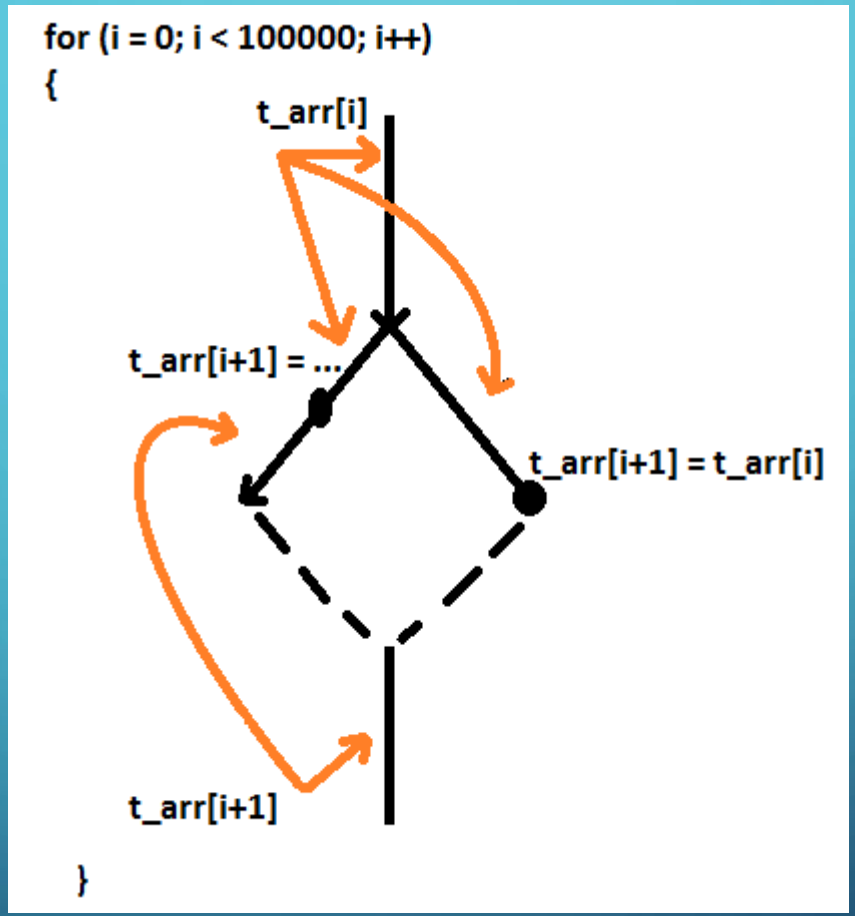

for (i = 0; i < 100000; i++)
{
t_arr[i]
t_arr[i + 1] = ...
t_arr[i+1]
}

```
for (i = 0; i < 100000; i++)
{
```

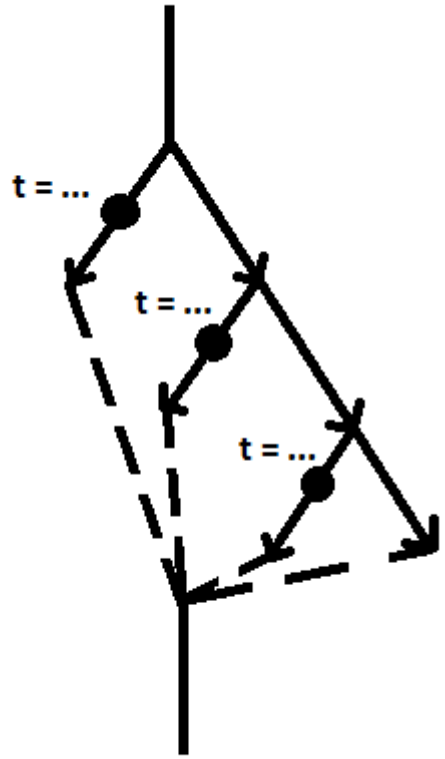



```
}
```


for (i = 0; i < 100000; i++)
{
t = ...
}


for (i = 0; i < 100000; i++)
{
t_arr[i]
t_arr[i+1] = ...
t_arr[i+1] = t_arr[i]
t_arr[i+1]
}

```
for (i = 0; i < 100000; i++)
{
```




```
}
```

ЧТО ИСПОЛЬЗУЕТСЯ АЛГОРИТМОМ?

Создание списка растягиваемых переменных

Используется граф информационных связей

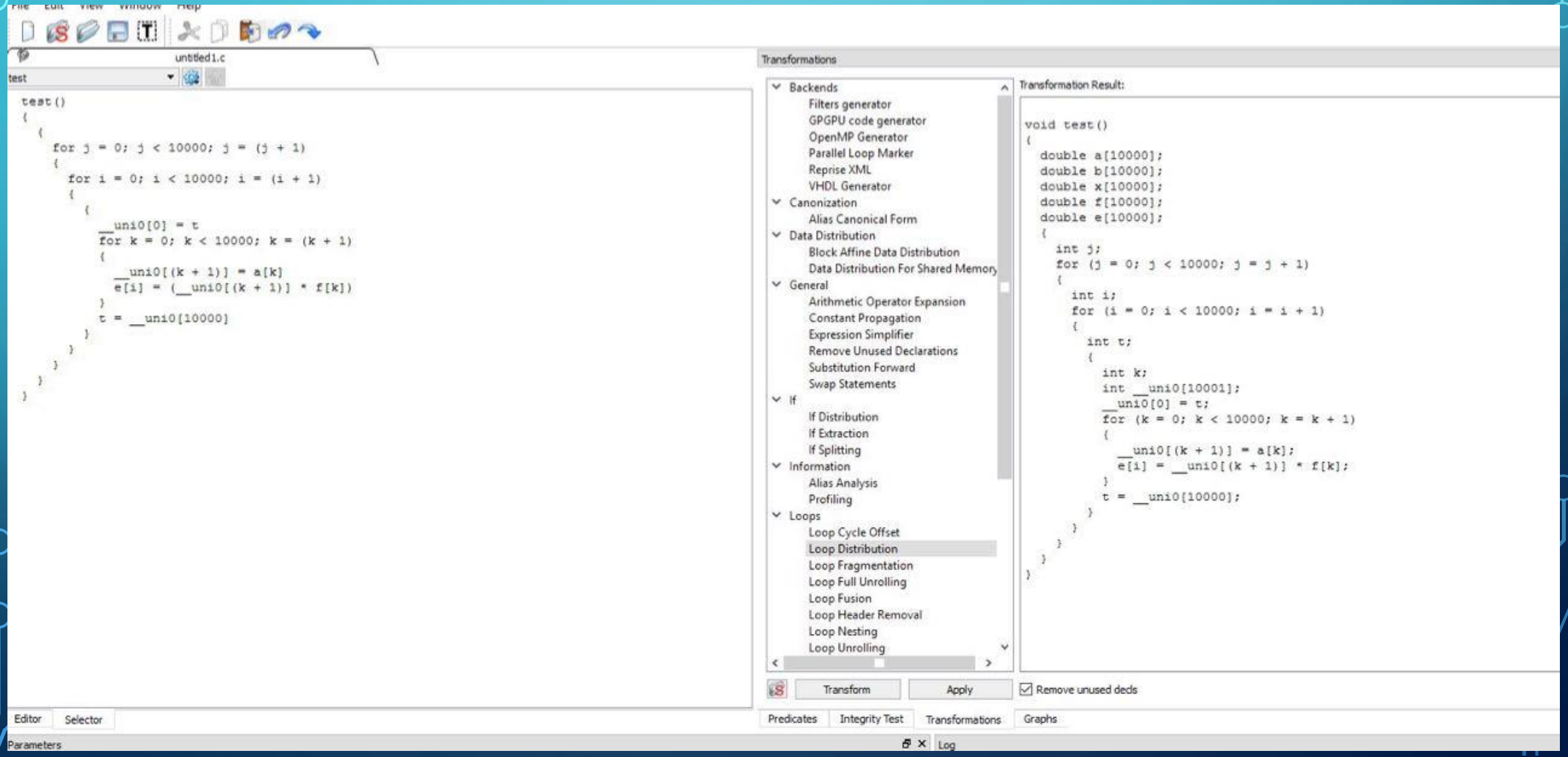
Растягивание скаляров

Используется обход в глубину до первого генератора в каждой ветке, граф информационных связей

ARRAY EXPANSION

```
...
for (t = 0; t < 100000; t++)
{
  for (i = 0; i < 100000; i++)
   {
      A[k][t] = E[i];
      T[i] = A[k] + C[i];
   }
}
...
```

ЗАКЛЮЧЕНИЕ


File Edit View Window Help
untitled1.c
test
test()
{
{
for j = 0; j < 10000; j = (j + 1)
{
for i = 0; i < 10000; i = (i + 1)
{
{
__uni0[0] = t
for k = 0; k < 10000; k = (k + 1)
{
__uni0[(k + 1)] = a[k]
e[i] = (__uni0[(k + 1)] * f[k])
}
t = __uni0[10000]
}
}
}
}
}
Editor
Selector
Parameters
Transformations
Backends
Filters generator
GPGPU code generator
OpenMP Generator
Parallel Loop Marker
Reprise XML
VHDL Generator
Canonization
Alias Canonical Form
Data Distribution
Block Affine Data Distribution
Data Distribution For Shared Memory
General
Arithmetic Operator Expansion
Constant Propagation
Expression Simplifier
Remove Unused Declarations
Substitution Forward
Swap Statements
If
If Distribution
If Extraction
If Splitting
Information
Alias Analysis
Profiling
Loops
Loop Cycle Offset
Loop Distribution
Loop Fragmentation
Loop Full Unrolling
Loop Fusion
Loop Header Removal
Loop Nesting
Loop Unrolling
Transform
Apply
Remove unused decls
Predicates
Integrity Test
Transformations
Graphs
Transformation Result:
void test()
{
double a[10000];
double b[10000];
double x[10000];
double f[10000];
double e[10000];
{
int j;
for (j = 0; j < 10000; j = j + 1)
{
int i;
for (i = 0; i < 10000; i = i + 1)
{
int t;
{
int k;
int __uni0[10001];
__uni0[0] = t;
for (k = 0; k < 10000; k = k + 1)
{
__uni0[(k + 1)] = a[k];
e[i] = __uni0[(k + 1)] * f[k];
}
t = __uni0[10000];
}
}
}
}
}
Log